5014: Session 3.Loops, if/else, packages in R

September 10, 2014

A quick note on packages

R packages can be downloaded and installed in order to increase functionality.

- Under the "packages" menu, click "install package(s)." (Must have internet connection.)
- Choose a location to download from.
- Choose a package from the list. Ex. Download "DAAG" to enable "pause" function.
- include 'library(*package name*)' in code to enable new commands and features.

When Google searching help files, package name comes after function name in parentheses.

For Loops

Sometimes you want to repeat same operation on an object a certain number of times:

- to fill in the elements of a vector/matrix,
- to do a particular analysis for a data set under different circumstances each time,
- to perform some of the statistical techniques such as: Bootstrap, Monte Carlo simulation, etc.

The number of repeats could be great or small.

The *for loop* syntax

The syntax is:

- *for('condition'){ code statement}*
- *('condition')* can be sequential numbers or elements of a vector:
  For example:
  1. *for (i in 1:10)* this statement will start the loop setting i=1, then move to i=2, so on until i=10.
  2. *for (i in c(1, 4 ,7, 12))* this statement will loop for i=1 as a first value for i, then move to i=4, then i=7, and finally i=12. and within curly brackets: *{'... statements...' }*

  Keep in mind that all the statements you want included in loop must be placed within the curly brackets: *{'... statements...' }*

For Loops

For example, look at the *Fisher's Iris data*

- Suppose you want to boxplot different variables sequentially, the following code is used:

```
attach(iris)
for (i in 1:4){
     boxplot(iris[,i]~Species,xlab='Species',
     ylab=paste(names(iris)[i]),
     main=paste('Boxplots of ',names(iris)[i],
     ' by species.'),col='slateblue',cex.lab=1.5)
     pause()
     }
```

paste() function concatenates vector elements and converts them into characters.

While loop

It is an useful control-flow when:

- Numerical optimization. Newton, Newton Raphson, or any other.
- As substitution of the *for loop* control-flow.

While loop syntax

while ('condition') { statements }.

- It evaluates the condition and remains in the loop until the condition is met.
- It also has to have the statements to evaluate within curly brackets.

If/else statement

Along with the *for loop* and *While loop*, the *If/else statement* is of great importance. When using *While loop* for

- To check the condition in order to leave the *While loop*.
- Or to create subset of your data set.
- These two criteria are checked using *If/else statement*.

Syntax:
if('condition'){ statements}
else { statements }

While loop and *If/else statement* Example

Suppose you want to find a solution for the equation:

$$f(x) = x^3 + x^2 + 1.$$

Its derivative is given by $f'(x) = 3x^2 + 2x$

The formula for Newton's method is: $x = x_0 - \frac{f(x_0)}{f'(x_0)}$. Using the following code you can find a solution for the eq $f(x) = 0$.

```
flag=0
xo=10
while (flag==0){
      x=xo-(xo^3+xo^2+1)/(3*xo^2+2*xo)
      tolerance =abs(x-xo)
      if (tolerance<=.00000001){
      flag=1
                }
      else{
                xo=x}
      }
```

Notes

- When using *While loop* be careful with the condition. A proper definition is needed, otherwise it will be in the loop indefinitely (*flag* variable in the example).
- In general R is slow for any loop.

Alternatives to loops

Since loops are slowly executed in R, you can use any of the following options:

1. lapply: *lapply(X, Function)*, returns an list object.
2. sapply: *sapply(X,Function)*, returns a data frame object.

Examples

To play with these two functions, let's call again the *Fisher's Iris data*. Suppose that you want to compute the mean and the variance of each continuous variable in the data.

- Using *lapply*,

  ```
  lapply(iris[,1:4],mean)
  ```

- Using *sapply*,

  ```
  sapply(iris[,1:4],var)
  ```